TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 24/31 pp331–344
DOI: 10.26599/TST.2023.9010106
Volume 30, Number 1, February 2025

Betweenness Approximation for Edge Computing with Hypergraph Neural Networks

Yaguang Guo, Wenxin Xie, Qingren Wang*, Dengcheng Yan, and Yiwen Zhang

Abstract: Recent years have seen growing demand for the use of edge computing to achieve the full potential of the Internet of Things (IoTs), given that various IoT systems have been generating big data to facilitate modern latency-sensitive applications. Network Dismantling (ND), which is a basic problem, attempts to find an optimal set of nodes that will maximize the connectivity degradation in a network. However, current approaches mainly focus on simple networks that model only pairwise interactions between two nodes, whereas higher-order groupwise interactions among an arbitrary number of nodes are ubiquitous in the real world, which can be better modeled as hypernetwork. The structural difference between a simple and a hypernetwork restricts the direct application of simple ND methods to a hypernetwork. Although some hypernetwork centrality measures (e.g., betweenness) can be used for hypernetwork dismantling, they face the problem of balancing effectiveness and efficiency. Therefore, we propose a betweenness approximation-based hypernetwork dismantling method with a Hypergraph Neural Network (HNN). The proposed approach, called “HND”, trains a transferable HNN-based regression model on plenty of generated small-scale synthetic hypernetworks in a supervised way, utilizing the well-trained model to approximate the betweenness of the nodes. Extensive experiments on five actual hypernetworks demonstrate the effectiveness and efficiency of HND compared with various baselines.

Key words: hypernetwork dismantling; Graph Neural Network (GNN); betweenness approximation; edge computing

1 Introduction

The Internet of Things (IoTs) has significantly changed the way we live in a variety of aspects[1, 2], including entertainment, agriculture, and manufacturing. In this situation, the recently emerged new computing paradigms have provided a beneficial complement to the traditional cloud-based systems. An example is edge computing, which can provide partial computing resources that are closer to the user or device side. Intuitively, the IoT is a heterogeneous service network, each with the ability to depict entities and their relations[3]. The IoT has been widely applied to model various application systems and is considered one of the major sources of data that facilitate modern latency-sensitive applications at the network edge, such as artificial intelligence, industrial automation, and smart transportation. Hence, edge computing is highly

• Yaguang Guo is with School of Management, Hefei University of Technology, Hefei 230009, China. E-mail: 2005800134@hfut.edu.cn.
• Wenxin Xie, Dengcheng Yan, and Yiwen Zhang are with School of Computer Science and Technology, Anhui University, Hefei 230601, China. E-mail: xiewxahu@foxmail.com; yanzhouahu.edu.cn; zhangyiwen@ahu.edu.cn.
• Qingren Wang is with Global Cognition and International Communication Laboratory, Anhui University, Hefei 230601, China. E-mail: wqr@ahu.edu.cn.
* To whom correspondence should be addressed.
Manuscript received: 2023-07-14; revised: 2023-09-11; accepted: 2023-09-27

demanded to achieve the full potential of the IoT.

However, as a basic problem in network science, Network Dismantling (ND)[4] aims to find a set of nodes whose removal will greatly destroy network connectivity. Due to the fact that network connectivity is highly related to the spread efficiency of information or even infectious diseases, ND has been widely applied in corresponding fields[5, 6]. Many researchers have proposed several dismantling methods to solve this problem. However, current methods[4, 7, 8] mainly focus on the traditional simple network that only considers the pairwise relation between two nodes. However, there are numerous higher-order groupwise relations among an arbitrary number of nodes in the real world, and traditional simple networks cannot model this kind of relation. Thus, the application of these methods is limited when facing groupwise relations.

Fortunately, a hypernetwork[9] has unique advantages in modeling such groupwise relations, that is why it has attracted increasing research attention. As shown in Fig. 1, a hyperedge in hypernetworks can naturally express the higher-order relations among an arbitrary number of nodes. Thus, in modeling real-world systems, a hypernetwork is more suitable than a traditional simple network, especially when dealing with groupwise relations. However, even if the higher-order relation modeling problem is solved by a hypernetwork, the existing dismantling methods still have limitations when it comes to hypernetwork applications. On the one hand, these methods cannot be directly applied due to the structural difference between a hypernetwork and a traditional simple network. On the other hand, although various centrality measures in a hypernetwork can be used for dismantling just like in a traditional network, both networks face the problem of balancing effect and efficiency. For example, betweenness centrality is very suitable for dismantling a network, but it requires

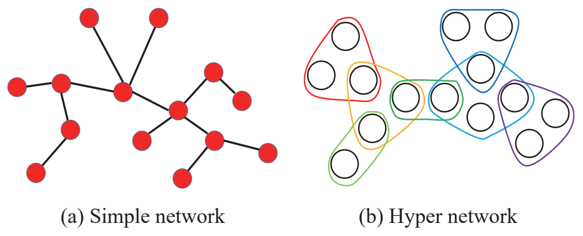


Fig. 1 Structural difference between a hypernetwork and a simple network.

complex calculations. Meanwhile, degree centrality is easy to calculate, but it has poor dismantling performance.

Recently, with the emergence of deep learning technology, researchers have attempted to utilize deep models to approximate complex centrality measures, such as betweenness[10] and closeness[11]. These measures can be approximated with less computational complexity within a limited error range. This means that the approximate centrality values are naturally utilized in various tasks, taking both effect and efficiency into consideration. Therefore, the present paper proposes a novel hypernetwork dismantling method based on deep learning technology, which we call "HND". This method adopts the Hypergraph Neural Network (HNN) to approximate betweenness centrality in hypernetworks and utilizes the approximate values to accomplish the task of hypernetwork dismantling. Specifically, HND first generates numerous small-scale synthetic hypernetworks and constructs betweenness ranking samples according to these. Then, an HNN-based betweenness ranking model is built, and the samples generated in the previous step are used to train this ranking model. Finally, the well-trained model is used to approximate the betweenness of all nodes in a given hypernetwork, and the hypernetwork is then dismantled based on the approximate betweenness values.

Our main contributions are summarized as follows:

- We design a betweenness approximation model based on an HNN. The model can be trained with numerous synthetic ranking samples and applied to real-world hypernetworks. With the help of deep learning's powerful representation ability, the trained model can well-approximate the betweenness of real-world hypernetworks with a lower computation complexity.
- We propose a novel hypernetwork dismantling method, called "HND", utilizing the betweenness approximation model to calculate the approximate betweenness that is adopted to achieve hypernetwork dismantling. Due to the approximation ability of HND, our proposed model can achieve performance close to betweenness with much lower computational complexity.
- We conduct extensive experiments on five real-world hypernetworks, and the results show that our proposed method outperforms the baselines. Moreover,

the experimental results confirm that the betweenness ranking model can approximate betweenness with less time consumption.

The remainder of the paper is organized as follows. Section 2 introduces the related works about DN and Graph Neural Networks (GNNs). Section 3 provides some necessary definitions. Section 4 introduces the proposed method in detail, while Section 5 describes the experimental settings and presents detailed analyses of the experimental results. Finally, Section 6 summarizes the whole paper and presents a potential direction for our future work.

2 Related Work

2.1 Network dismantling

ND[4] is a problem that aims to find an optimal set of nodes whose removal will greatly destroy network connectivity. It is an NP-hard graph combinatorial optimization problem with an exact solution that is difficult to obtain. Thus, researchers have attempted to find approximate solutions and propose various ND methods.

Generally, current ND methods can be divided into three classes. The first class is based on centrality measures in which nodes are selected greedily according to their centrality measures. However, these methods typically face the problem of balancing effectiveness and efficiency. Local centrality measures (e.g., degree centrality) are easy to calculate but cannot achieve good dismantling performance, while global centrality measures (e.g., betweenness centrality and closeness centrality) have good dismantling ability but have a high computation complexity. To take both effectiveness and efficiency into consideration, some centrality measures utilizing mesoscopic network structures have been proposed. For example, the Collective Influence (CI), proposed by Morone and Makse[12] considers both the dismantling effectiveness and efficiency by flexibly balancing globality and locality through a tuning hyperparameter.

The second class of ND methods consists of heuristic methods that work by dismantling a network using multiple heuristic steps. For example, Braunstein et al.[4] proposed a three-step method called “MinSum”, which dismantles a network by decycling, tree-breaking, and node reinsertion. Similar to MinSum, CoreHD[13] and BPD[14] also adopt this framework but have differences in details. Moreover, Ren et al.[7] proposed the GND algorithm to consider the case of weighted nodes.

The third class of ND methods is based on deep learning, wherein researchers aim to achieve better dismantling effectiveness with the help of the powerful ability of deep learning. Specifically, Fan et al.[8] proposed FINDER based on deep reinforcement learning. This method trains an agent to perform dismantling exercises on numerous small-scale synthetic networks, and is then applied to real large-scale networks. Meanwhile, Grassia et al.[15] proposed the GDM method that trains a GNN ranking model using ground-truth dismantling sequences, and then applies it to real ND. Recently, with the surge of hypernetworks, Yan et al.[16] proposed a dismantling method suitable for hypernetworks based on deep reinforcement learning called HITTER. Generally, current works mainly focus on traditional simple networks but have so far ignored hypernetworks. Thus, in the present paper, we attempt to solve the hypernetwork dismantling problem via deep learning technology.

2.2 GNNs

As a kind of network embedding method, GNNs can map nodes into low-rank vectors according to specific tasks. Various GNNs based on the most basic neighbor information aggregation mechanisms have been proposed and applied to many fields. Among these, the Graph Convolution Network (GCN)[17, 18] is the most common method. Through neighbor information aggregation, feature linear transformation, and nonlinear activation, node embedding can be obtained and applied to various downstream tasks, such as node classification and line prediction. Other GNNs following this framework have been proposed as well. For example, in the Graph Attention Network (GAT)[19], different neighbors have varying levels of importance to the target nodes in the step of neighbor aggregation. GAT has also introduced the attention mechanism to make target nodes adaptively aggregate information. Hamilton et al.[20] proposed the inductive GraphSAGE, which can infer the embeddings of nodes unseen in the training stage. Due to their powerful expression ability, GNNs have been applied to various fields, such as recommendation systems[21–23] and user profiles[24, 25]. However, the core neighbor information aggregation mechanism in GNN relies on pairwise interactions. Therefore, because of the structural differences with traditional simple networks, these GNNs cannot be directly applied to hypernetworks. To solve this problem, researchers have proposed various methods to extend the traditional GNNs to hypernetworks. For example, Feng et al.[26] proposed HGNN, which applies GCN to hypernetworks by transforming hypernetworks into simple networks in accordance with clique expansion. Similarly, Yadati et al.[27] transformed hypernetworks into simple networks by breaking hyperedges into pairwise edges via specific rules and then directly applying GCN to these.

Unlike the abovementioned methods, the UniGNN proposed by Huang and Yang[28] features a neighbor information aggregation mechanism that is suitable for hypernetworks. Through the aggregation paths of nodes to hyperedges and hyperedges to nodes, various traditional GNNs, such as GCN, GAT, and GIN[29], can be transferred to hypernetworks.

In this paper, we adopt the HNN to accomplish the betweenness approximation task. Although some works have attempted to approximate betweenness via GNNs, they are not suitable for hypernetworks. Therefore, an HNN for hypernetwork betweenness approximation is necessary.

3 Preliminary

In this section, we briefly introduce some related concepts.

Definition 1 (**Hypernetwork[9]**) A hypernetwork is defined as $G=(V,E)$, where $V=\{v_1,v_2,\ldots,v_N\}$ denotes the node set in the hypernetwork, and $N=|V|$ is the number of nodes. The $E=\{e_1,e_2,\ldots,e_M\}$ is the set of hyperedges, and $M=|E|$ denotes the number of hyperedges. Each hyperedge e is defined as $e\subseteq V$ and $e\neq\phi$.

For each hyperedge e in the hypernetwork, the size of e denotes the number of nodes contained in it. Obviously, given that the size of hyperedges is flexible, the hypernetwork can model both pairwise and groupwise interactions. This hypernetwork becomes a traditional simple network when the size of all hyperedges in a hypernetwork equals 2. Therefore, a traditional simple network can be seen as a specific form of hypernetwork.

Definition 2 (**Incidence matrix[30]**) The incidence matrix of a hypernetwork is a matrix $\boldsymbol{H}\in\mathbf{R}^{N\times M}$. Each element of $\boldsymbol{H}$ is given below:

$$\boldsymbol{H}_{v_i,e_j}=\begin{cases}1, & v_i\in e_j;\\0, & \text{otherwise}\end{cases}\tag{1}$$

Similar to the adjacent matrix in traditional simple networks, the incidence matrix can be used to express relations between nodes and hyperedges in a hypernetwork. Figure 2a shows a sample hypernetwork and its incidence matrix.

Definition 3 (**Hyperdegree[30] and degree[9]**) For each node v_i in hypernetwork G, the hyperdegree of v_i is defined as the number of hyperedges containing node v_i,

$$\text{hdeg}(v_i)=\sum_{e_j\in E}\boldsymbol{H}_{v_i,e_j}\tag{2}$$

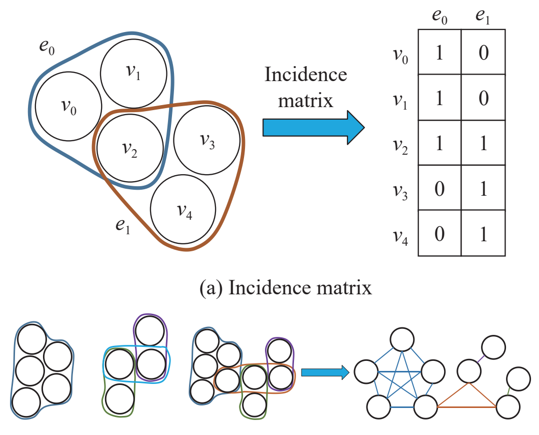
The degree, namely $\deg(v_i)$, denotes the number of nodes adjacent to the v_i,

$$\deg(v_i)=\sum_{v_j\in V}(\boldsymbol{H}\cdot\boldsymbol{H}^{\mathrm{T}})_{v_i,v_j}-\text{hdeg}(v_i)\tag{3}$$

Given that the dismantling problem is highly related to network connectivity, we define the connectivity of a hypernetwork as follows.

Definition 4 (**Hypernetwork connectivity**) The connected component containing the highest number of hyperedges in hypernetwork G is called the Giant Connected Component (GCC), and the connectivity of G, denoted as connectivity (G), is defined as the ratio of the number of nodes in GCC (which is denoted as $|V_{\text{GCC}}|$) to the number of nodes in G (which is denoted as $|V_G|$),

$$\text{connectivity}(G)=\frac{|V_{\text{GCC}}|}{|V_G|}\tag{4}$$



(a) Incidence matrix

(b) Connectivity (c) 2-section network

Fig. 2 Hypernetwork and its corresponding definitions.

Unlike in a traditional simple network, our definition of hypernetwork connectivity is related to the connections between hyperedges. According to Berge[31], a connected hypernetwork relies on its relations among hyperedges, not nodes. Moreover, selecting the connected component containing the highest number of nodes as the GCC is also reasonable. However, the connectivity defined in this way is likely to be influenced by those huge hyperedges. As shown in Fig. 2b, selecting the left-connected component cannot reflect the property of hypernetwork connectivity, and this move also violates the purpose of dismantling.

Definition 5 (**2-section network[30]**) A 2-section network is a traditional simple network transformed from a hypernetwork. For each hyperedge e in a hypernetwork G, the groupwise interaction can be transformed into multiple pairwise interactions through linking each two nodes in e. Furthermore, the 2-section network is obtained when all hyperedges are transformed over.

As shown in Fig. 2c, each hypernetwork can be transformed into a simple network. In this way, various methods designed for a simple network can be applied to a hypernetwork.

4 Proposed Method: HND

4.1 Overall framework

In this section, we introduce our proposed method in detail. Generally, our proposed HND method can be divided into three steps.

Step 1: To supply training samples for the subsequent model, we adopt a hypernetwork generator to generate numerous small-scale synthetic hypernetworks and calculate each node's betweenness value in these networks. Then, training samples are constructed in accordance with the ground-truth betweenness values (as shown in Section 4.2).

Step 2: Based on the HNN, we build a node betweenness approximation model in a hypernetwork. This model can be applied to approximate the nodes' betweenness in a hypernetwork based on the network structure and predict the nodes' approximation betweenness values (as shown in Section 4.3).

Step 3: By combining the node ranking samples and approximation betweenness values, we build the pairwise ranking loss to optimize the parameters in the ranking model. After multiple iterations, the model can be used to approximate the nodes' betweenness in real-world hypernetworks; it can also be further applied to hypernetwork dismantling (as shown in Section 4.4).

4.2 Training sample generation

In this step, numerous node pair samples are generated according to their betweenness values. Thus, a synthetic hypernetwork generator is needed to generate small-scale hypernetworks. Two synthetic hypernetwork generators have been recently proposed, namely, HyperPA[32] and HyperFF[33]. The main difference between them is that HyperPA generates hypernetworks according to a predefined distribution from real-world hypernetworks, while HyperFF adopts two hyper-parameters (i.e., burning probability p and expanding probability q) to tune the density of synthetic hypernetworks. Furthermore, compared with HyperPA, HyperFF is more flexible and can produce more generalized hypernetworks. Thus, HyperFF is adopted in the present study to accomplish this task.

Once the generator is chosen, many small-scale synthetic hypernetworks can be generated. For each synthetic hypernetwork G, we transform it into its 2-section network, after which we calculate the corresponding betweenness value $\boldsymbol{B}$ (where $\boldsymbol{B}_{v_i}$ denotes the betweenness value of node v_i). Then, multiple node pairs are sampled to construct ranking instances. For two random nodes v_i and v_j, instances (v_i, v_j, l_{v_i,v_j}) can be constructed,

$$l_{v_i,v_j} = \begin{cases} 1, & \boldsymbol{B}_{v_i} > \boldsymbol{B}_{v_j}; \\ 0, & \boldsymbol{B}_{v_i} < \boldsymbol{B}_{v_j} \end{cases} \tag{5}$$

where l_{v_i,v_j} represents the difference between v_i and v_j. Notably, ranking instances cannot be constructed when two nodes' betweenness values are equal. Therefore, this case should be avoided when sampling node pairs.

Through the method described above, each synthetic hypernetwork can generate one corresponding ranking sample set, denoted as S_G, after which the training samples space Ω is also obtained, as shown below:

$$S_G = \left\{(v_0^G, v_1^G, l_{v_0,v_1}^G), (v_1^G, v_2^G, l_{v_1,v_2}^G), \ldots\right\} \tag{6}$$

$$\Omega = \{S_{G_1}, S_{G_2}, \ldots\} \tag{7}$$

4.3 Betweenness approximation model

To approximate the betweenness of a node in a hypernetwork, we build an HNN-based ranking model. For each ranking sample set S_G, our model first obtains the nodes' embeddings via an HNN. Then, the nodes' embeddings are fed into a fully connected neural network to obtain the approximation betweenness values. The detailed process is introduced below.

For a given hypernetwork G, we first map the nodes into their dense vector form with an HNN. However, conventional transductive HNNs, such as HGNN and HyperGCN, are not applicable in our framework because the training process is conducted on a large number of small-scale synthetic hypernetworks. Then, the trained model is applied to real-world hypernetworks. Thereafter, we choose the inductive HNN called HyperSAGE[16], which we proposed in our previous work. Generally, HyperSAGE contains two levels of information aggregation, i.e., hyperedge and node level aggregation.

In the step of hyperedge level aggregation, the hyperedge features (denoted as $\boldsymbol{A}^l$) are first obtained according to the nodes' features, as shown below:

$$\boldsymbol{A}^l = \text{softmax}\,(\boldsymbol{H}^{\mathrm{T}} \odot (\boldsymbol{X}^l \boldsymbol{W}_1)^{\mathrm{T}}) \tag{8}$$

$$\boldsymbol{Y}^l = \boldsymbol{A}^l \boldsymbol{X}^l \tag{9}$$

where $\boldsymbol{X}^l \in \mathbf{R}^{N\times D_l}$ and $\boldsymbol{Y}^l \in \mathbf{R}^{M\times D_l}$ are the nodes' and hyperedges' embeddings in the l-th layer, respectively (D_l denotes the embedding dimension in the l-th layer). In addition, $\boldsymbol{W}_1 \in \mathbf{R}^{D_l\times 1}$ refers to the trainable parameters, and $\odot$ denotes the operation of the element-wise product. Function softmax is used to normalize the aggregation weight. Notably, we set $\boldsymbol{X}^0 = \mathbf{1}$ due to the lack of initial node features $\boldsymbol{X}^0$. Based on the view of information propagation, nodes with high betweenness values—as hubs of multiple shortest paths—have more powerful information propagation ability than those with low betweenness values. Thus, the initial node feature $\mathbf{1}$ can be regarded as the initial information, and each layer of HNN is a kind of information propagation. After multiple layers, nodes' embeddings can reflect their ability of information propagation, thus enabling their further application to betweenness approximation. Once the hyperedges' embeddings are obtained, the hyperedge level aggregation can be performed as shown below:

$$\boldsymbol{Y}^{l+1} = f\,([\boldsymbol{Y}^l \boldsymbol{W}_2 \| (\boldsymbol{H}^{\mathrm{T}} \boldsymbol{H} \boldsymbol{Y}^l \boldsymbol{W}_3)]\boldsymbol{W}_4) \tag{10}$$

where $\boldsymbol{W}_2$, $\boldsymbol{W}_3 \in \mathbf{R}^{D_l\times D_{l+1}}$ and $\boldsymbol{W}_4 \in \mathbf{R}^{2D_{l+1}\times D_{l+1}}$ are all trainable parameters. $\|$ denotes the operation of matrix concatenation, and the nonlinear activation function $f()$ is specified as ReLU.

The next step is node-level aggregation. In this step, node aggregate information from their adjacent hyperedges is obtained, as shown below:

$$\boldsymbol{X}^{l+1} = f\,([\boldsymbol{X}^l \boldsymbol{W}_5 \| \boldsymbol{H}\boldsymbol{Y}^{l+1} \boldsymbol{W}_6]\boldsymbol{W}_7) \tag{11}$$

where $\boldsymbol{W}_5 \in \mathbf{R}^{D_l \times D_{l+1}}$, $\boldsymbol{W}_6 \in \mathbf{R}^{D_{l+1} \times D_{l+1}}$, and $\boldsymbol{W}_4 \in \mathbf{R}^{2D_{l+1}\times D_{l+1}}$ are trainable parameters.

Once multiple layers are chained in HyperSAGE, the nodes' final embeddings $\boldsymbol{X}^L$ are obtained (L denotes the total number of layers in HyperSAGE). By feeding them into a fully connected neural network, the approximated betweenness values $\hat{\boldsymbol{B}}$ of the nodes can be calculated by

$$\hat{\boldsymbol{B}} = f\,(\boldsymbol{X}^L \boldsymbol{W}_8 + b) \tag{12}$$

where $\boldsymbol{W}_8 \in \mathbf{R}^{D_L\times 1}$ and $b \in \mathbf{R}$ are trainable parameters.

Thus, the approximated betweenness values of nodes are obtained through the above steps. According to the values, a hypernetwork can be dismantled by greedily removing nodes with the highest approximated betweenness values. Upon the removal of each batch of nodes, the approximated betweenness values of the residual nodes are recalculated due to the structural changes of the hypernetwork. Betweenness value approximation and node removal are then conducted repeatedly until the scale of GCC in the hypernetwork decreases to a threshold value or when all nodes are removed.

4.4 Optimization

To optimize the betweenness approximation model, node ranking samples generated in Section 4.2 are utilized to update all trainable parameters in the model. Thus, we build the pairwise ranking loss as follows.

For each samples set S_G, the Bayesian Personalized Ranking (BPR)[34] loss of each instance $(v_i^G, v_j^G, l_{v_i,v_j}^G)$, denoted as loss_{v_i,v_j}^G, is calculated,

$$\hat{l}_{v_i,v_j}^G = \text{sigmoid}\left(\hat{\boldsymbol{B}}_{v_i}^G - \hat{\boldsymbol{B}}_{v_j}^G\right) \tag{13}$$

$$\text{loss}_{v_i,v_j}^G = -l_{v_i,v_j}^G \log\left(\hat{l}_{v_i,v_j}^G\right) - \left(1 - l_{v_i,v_j}^G\right)\log\left(\hat{l}-_{v_i,v_j}^G\right) \tag{14}$$

where $\hat{\boldsymbol{B}}_{v_i}^G$ and $\hat{\boldsymbol{B}}_{v_j}^G$ denote the approximated betweenness values of node v_i and node v_j in hypernetwork G, respectively, $\hat{l}_{v_i,v_j}^G$ denotes the value calculated by sigmoid. In accordance with the pairwise loss of a single instance, the total loss of the whole sample Ω can be obtained as below.

$$\text{loss} = \frac{1}{|\Omega|}\sum_{S_G\in\Omega}\frac{1}{|S_G|}\sum_{\left(v_i^G, v_j^G, l_{v_i,v_j}^G\right)\in S_G} \text{loss}_{v_i,v_j}^G \tag{15}$$

Next, we use gradient decrease[35] to update all trainable parameters in our model. Through multiple iterations of parameter updating, the loss of the model will converge. Then, the model can be used to approximate nodes' betweenness in real-world hypernetworks.

4.5 Time complexity

In the process of inference, the time complexity of HND is mainly due to hypernetwork embedding (i.e., HyperSAGE) and the approximation function (i.e., the fully connected neural network). For the former part, there are two steps in each layer of HyperSAGE. (1) In hyperedge level aggregation, the time complexities of attention mechanism and information propagation are both $O(M\times D)$; (2) In node-level aggregation, the time complexity of information propagation is $O(N\times D)$. Therefore, the time complexity of HyperSAGE is $O((N+2\times M)\times L\times D)$. For the latter part, the time complexity of the fully connected layer is $O(N\times D)$. Thus, the inference complexity of HND is $O((N+2\times M)\times L\times D+N\times D)$. While for exact betweenness centrality calculation, its time complexity for unweighted networks is $O(N\times M)$. Given that L and $D\ll N$ and M in most real-world scenarios, our proposed HND has a linear time complexity and is more practically applicable than exact betweenness centrality calculation, which is showed in Algorithm 1.

5 Experiment

In this section, we conduct extensive experiments to verify the effectiveness of our proposed method.

5.1 Experimental datasets and settings

5.1.1 Experimental datasets

We collect five real-world hypernetworks to evaluate our proposed method. Brief introductions about these datasets are listed below.

- **Cora-co-authorship**[27] This dataset contains scientific papers published in the field of machine learning. We construct a hypernetwork by taking authors as nodes and co-author relations as hyperedges.
- **Citeseer**[27] This dataset contains scientific papers in six fields, along with their citation relations. To construct the hypernetwork, we map the papers and citation relations as nodes and hyperedges, respectively.
- **MAG**[36] This dataset contains scientific papers and authors from the field of history in Microsoft Academic Graph. Similar to the Cora-co-authorship dataset, we map papers as nodes and co-author relations as hyperedges.
- **Pubmed**[27] This dataset contains scientific papers from the field of diabetes, along with their citation relations. Given that Pubmed is the same as Citeseer, the method of constructing hypernetworks in Citeseer is also suitable for Pubmed.
- **NDC**[37] This dataset contains information about many kinds of drugs, each consisting of multiple substances. The hypernetwork can be constructed by considering substances and drugs as nodes and hyperedges, respectively.

For these datasets, we conducted some preprocessing steps. Due to the dismantling problem being concentrated on the connectivity of the network, a disconnected network is likely to disturb the experimental results. Therefore, for each original hypernetwork, we only select their GCCs as the initial

Algorithm 1 Training process of HND

```
Require: Scale range of synthetic hypernetworks ($N_{\min}$, $N_{\max}$), burning propabality range ($p_{\min}$, $p_{\max}$) in HyperFF, expanding propabality range ($q_{\min}$, $q_{\max}$) in HyperFF, synthetic hypernetwork number $J$, max iteration number $I$, HyperSAGE layer number $L$, embedding dimension $D$, and node pairs sample ratio $r$
Ensure: Betweenness approximation model parameter $\Theta$
1: Initialize model parameter $\Theta$ randomly;
2: Initialize sample set $\Omega = \{\}$
3: for $n$ = 1 to $J$ do
4:   Take random values $p$, $q$, $N$ in range [$p_{\min}$, $p_{\max}$], [$q_{\min}$, $q_{\max}$], and [$N_{\min}$, $N_{\max}$], respectively;
5:   Generate a hypernetwork $G$ which has $N$ nodes by HyperFF with hyperparameters $p$ and $q$;
6:   Calculate the exact betweenness values $\hat{B}^G$ of $G$'s corresponding 2-section network;
7:   Sample $\lceil rN \rceil$ node pairs randomly;
8:   Generate ranking samples $S_G$ according to Eqs. (5) and (6);
9:   Add $S_G$ into sample set $\Omega$;
10: end for
11: for $i$ = 1 to $I$ do
12:   for $S_G \in \Omega$ do
13:     Embed nodes in hypernetwork $G$ according to Eqs. (8)–(11);
14:     Calculate nodes' approximated betweenness values according to Eq. (12);
15:     Calculate pairwise loss according to Eq. (14);
16:     Update model parameters $\Theta$ through gradient decrease;
17:   end for
18: end for
19: return model parameter $\Theta$
```

hypernetwork waiting to be dismantled. Some statistics of hypernetworks after preprocessing are summarized in Table 1.

5.1.2 Baselines

We select various baselines to evaluate the dismantling performance of our method, and their introductions are presented below:

- **Highest Degree Adaptive (HDA)** This method dismantles a network according to degree. Nodes with the highest degree will be removed in each removal step. After each removal step, the nodes' degrees will be recalculated due to the changes in network structure.
- **Highest HyperDegree Adaptive (HHDA)** This method dismantles a hypernetwork according to hyperdegree. Similar to HDA, nodes with the highest hyperdegree will be removed in each step, and their hyperdegrees will be updated upon removal.
- **Collective Influence (CI)[12]** This method removes nodes according to their CI values, and the way of calculating the CI value for each node is given below:

$$\mathrm{CI}_{v_i} = (\deg(v_i)-1)\sum_{v_j \in \mathrm{Nei}_k(v_i)} (\deg(v_j)-1) \tag{16}$$

where $\mathrm{Nei}_k(v_i)$ denotes the k-hop neighbors of node v_i, and in this paper, we set k to be 2. The node with the highest CI value will be removed in each step, and the CI values of residual nodes will also be recalculated.

- **GND[7]** GND first computes the network's weighted Laplacian matrix, which is utilized to obtain nodes' eigenvector through spectrum approximation. Then, we split the nodes into two groups according to their eigenvector. Finally, we used the weighted vertex cover algorithm to select the nodes to be removed.
- **FINDER[8]** This method is based on deep reinforcement learning. It builds an agent and makes it to perform dismantling exercises on many small-scale synthetic networks to optimize the agent's dismantling strategy. Once its strategy converges, the agent can be used to dismantle real-world networks.
- **SubTSSH[38]** This method is designed to solve the problem of selecting target nodes set in hypernetworks. By iteratively conducting node removal and influence propagation, this can be used for hypernetwork dismantling.
- **HITTER[16]** This method is specifically designed for hypernetwork dismantling based on deep reinforcement learning. First, it builds an agent to do trial-and-error on many small-scale synthetic hypernetworks. After the agent's dismantling strategy is well optimized, it can be utilized to dismantle real-world hypernetworks.

In the abovementioned baselines, HDA, HHDA, CI, and SubTSSH are methods based on centrality measures. GND is a recently proposed heuristic method, while FINDER and HITTER are methods based on deep learning. Generally, these baselines cover several commonly used dismantling methods. Moreover, some of these baselines (HDA, CI, GND, and FINDER) cannot be directly applied to hypernetworks. Therefore, we transform the hypernetworks into their corresponding 2-section networks when applying these methods.

5.1.3 Metrics

We adopt Accumulated Normalized Connectivity (ANC)[39] to evaluate the performance of hypernetwork dismantling. The calculation of ANC is given below:

$$\mathrm{ANC}(\kappa) = \frac{1}{K}\sum_{k=1}^{K} \frac{\mathrm{connectivity}\,(G\backslash\{v_1, v_2, \ldots, v_k\})}{\mathrm{connectivity}\,(G)} \tag{17}$$

where $\kappa = \{v_1, v_2, \ldots, v_K\}$ is the node removal sequence obtained by various dismantling methods, and $G\backslash\{v_1, v_2, \ldots, v_k\}$ denotes the residual hypernetwork after removing nodes $\{v_1, v_2, \ldots, v_k\}$ from G. Given a node removal sequence κ, a low ANC value means this sequence can dismantle the network effectively.

5.1.4 Experimental settings

In the step of sample generation, the minimal scale of synthetic hypernetwork $N_{\min}$ is set to 100, and the maximal scale $N_{\max}$ is set to 150. The burning probability range $(p_{\min}, p_{\max})$ and expanding probability range $(q_{\min}, q_{\max})$ are all fixed to (0.1, 0.4).

Table 1 Statistics of datasets.

Dataset	Number of nodes	Number of hyperedges	Average node hyperdegree	Average hyperedge size
Cora-co-authorship	1676	463	1.66	6.00
Citeseer	1019	626	2.23	3.63
MAG	1669	784	1.59	3.38
NDC	3065	4533	13.57	9.17
Pubmed	3825	5432	7.45	5.25

The number of synthetic hypernetworks J is set to 1000, and for each hypernetwork we randomly sample $\lceil 0.9N \rceil$ node pairs as training samples. In the betweenness approximation model, the number of layers of HyperSAGE is set to 4, and the embedding dimension is set to 32. In the training process, the learning rate of the model is set to 0.005. Furthermore, we generate 50 synthetic hypernetworks as a validation dataset to verify the model training. We also adopt an early stopping mechanism to avoid model over-fitting, and its patience is fixed at 100.

5.2 Experimental results and analyses

5.2.1 Overall performance

We apply the proposed HND and baselines to dismantle those real-world hypernetworks in datasets, and their performance is summarized in Table 2. As shown in the results, we can easily conclude that our proposed HND approach has a significant dismantling performance than the baselines in the majority of datasets. This finding indicates the effectiveness of the HND in approximating betweenness for hypernetwork dismantling with the advantage of reducing computational complexity. We present further details of the dismantling process in Fig. 3. The detailed ANC curves in Fig. 3 show that, for most hypernetworks, the removal of only a small portion of nodes will significantly destroy their connectivity. Under the same removal budget (i.e., removing fraction), our proposed HND often fragments the hypernetwork into the smallest connected components.

Notably, the overall performance of the dismantling methods designed for traditional simple networks, i.e., HDA, CI, GND, and FINDER, is generally poorer than those designed specifically for hypernetworks, i.e., HHDA, SubTSSH, and HITTER. This is because hypernetworks must be transformed to their 2-section forms before applying dismantling methods designed for traditional simple networks, and this kind of transformation introduces some noisy structures, such as dense cliques. Moreover, to our best knowledge, using FINDER as the state-of-the-art method for traditional ND performs even worse than the other traditional simple ND methods. This could be due to the fact that it utilizes the BA model[40] for synthetic network generation in the training process. However, the BA model is a generative model for traditional simple networks, and it cannot model the structure of hypernetwork.

5.2.2 Impact of the number of embedding layers

HND relies on a multiple-layer HNN to extract structural information into node embeddings, which are then utilized for betweenness approximation. Different numbers of layers indicate different levels of awareness abilities of global hypernetwork structure; thus, the number of layers L has a vital influence on hypernetwork dismantling performance. The impact of the number of embedding layers on each dataset is shown in Fig. 4. From this figure, we can conclude that an excessively small or large L value will reduce the hypernetwork dismantling performance. The reason may be that too few layers will limit the information propagation of HNN. In turn, this phenomenon would prevent the node embeddings from preserving enough global structure information, while too many layers will excessively smoothen the node embeddings and make nodes less distinguishable. In addition, too many layers can bring extra computational costs. Thus, taking both effectiveness and efficiency into account, it is necessary to choose a proper number of embedding layers in real-world applications.

5.2.3 Analysis of model efficiency

The motivation for proposing HND is to reduce the high computational complexity of global structure-based centrality (such as betweenness) while preserving its effectiveness in hypernetwork dismantling.

First, the inference efficiency of HND is compared against HITTER. The exact betweenness centralities under different hypernetwork scales and the time

Table 2 Overall performance in terms of ANC.

Dataset	Method							
	HDA	CI	GND	FINDER	HHDA	SubTSSH	HITTER	HND
Cora-co-authorship	0.1564	0.1181	0.1111	0.4068	0.0977	0.1267	0.0792	**0.0713**
Citeseer	0.0930	0.0915	0.2528	0.1109	0.0788	0.0965	0.0607	**0.0592**
MAG	0.0261	0.0238	0.0335	0.0410	0.0195	0.0226	**0.0130**	0.0161
Pubmed	0.3933	0.3930	0.3606	0.4809	0.3831	0.4038	0.3529	**0.3344**
NDC	0.2608	0.2623	0.4372	0.4804	0.2374	0.2666	0.2209	**0.2144**

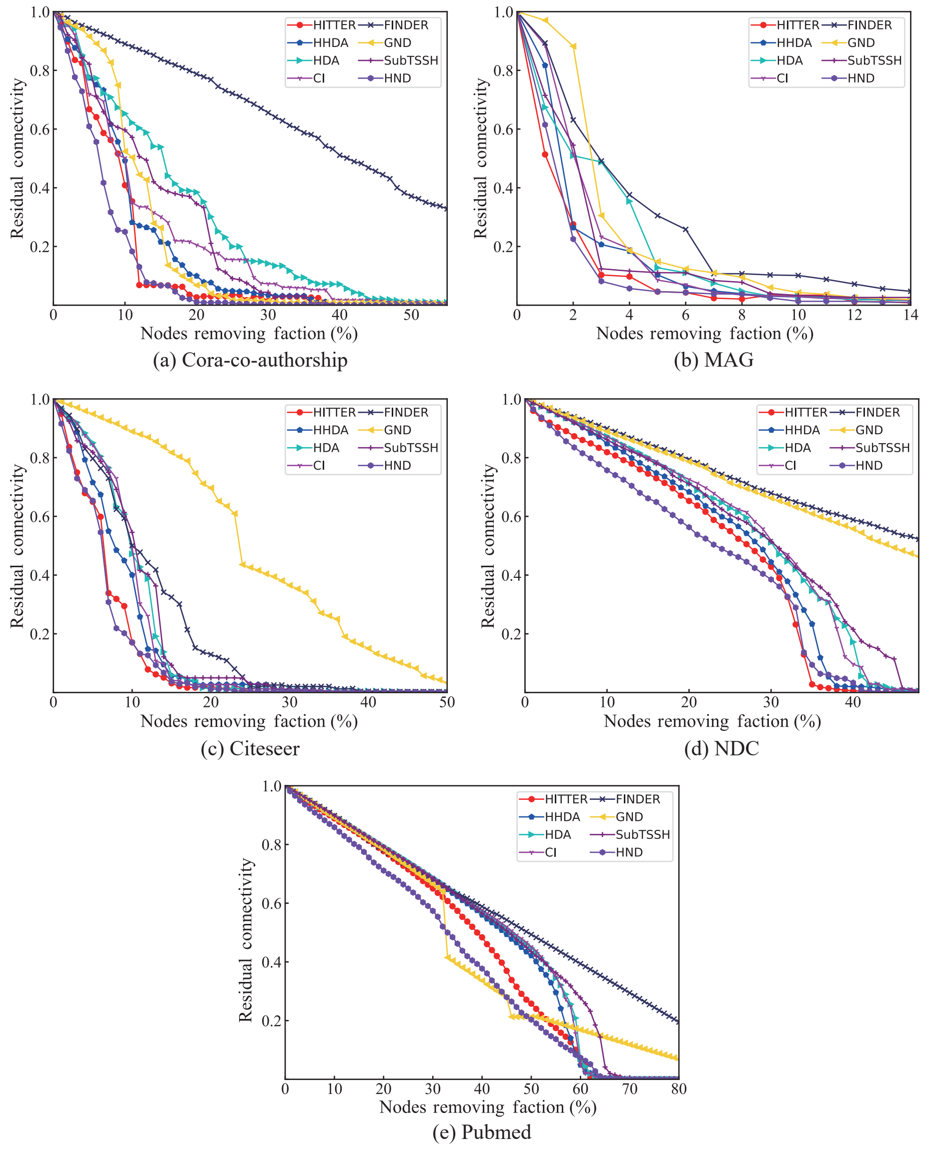


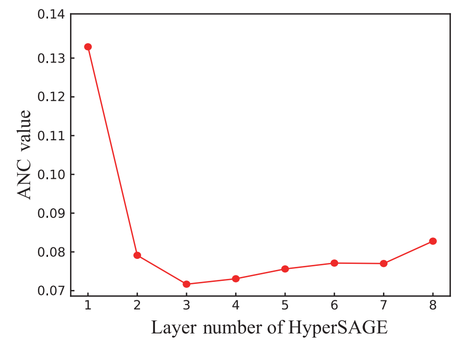
Fig. 3 Detailed dismantling curve.

consumption of each calculation in these methods are shown in Fig. 5. As shown in Fig. 5, the inference times of HND and HITTER increase linearly with the hypernetwork scale, while the inference time of the exact betweenness centrality grows quadratically. This indicates a potential application of our proposed HND to large-scale hypernetworks.

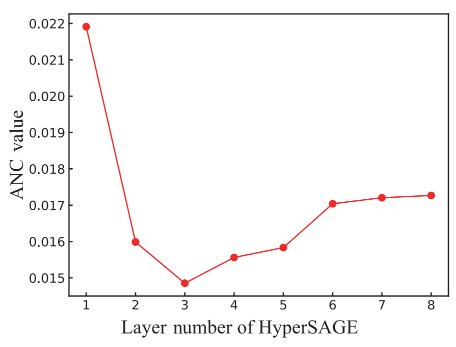
Moreover, given that HND and HITTER are deep learning based hypernetwork dismantling methods, we also compare their training efficiencies. From the results shown in Table 3, we can see that the training of HITTER usually needs tens of thousands of iterations, while that of HND only needs hundreds of iterations. The training time of HND is also obviously less than that of HITTER. The core reason behind this is the difference in training mode. In particular, the reinforcement learning adopted by HITTER trains the model with exploration and exploitation, and a huge amount of time is needed to explore effective dismantling actions. In comparison, the supervised learning adopted by our HND directly guides the model training with effective dismantling actions. Thus, HND can significantly decrease the training time than HITTER.

6 Conclusion

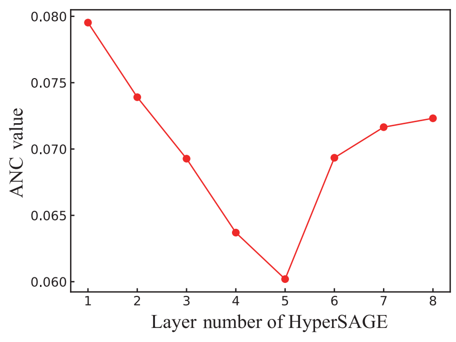
In this paper, we propose a betweenness approximation method based on hypernetwork dismantling called HND. To achieve betweenness approximation in a supervised manner, we utilize an HNN to preserve



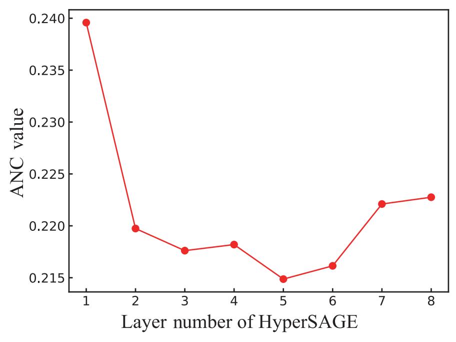
(a) Cora-co-authorship



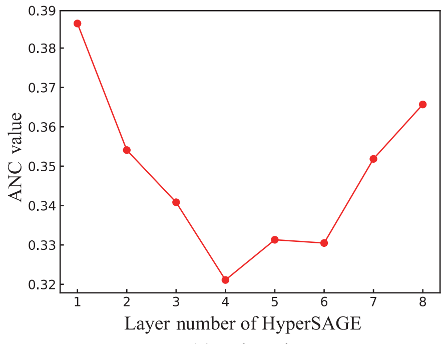
(b) MAG



(c) Citeseer



(d) NDC



(e) Pubmed

Fig. 4 Impact of the number of embedding layers.

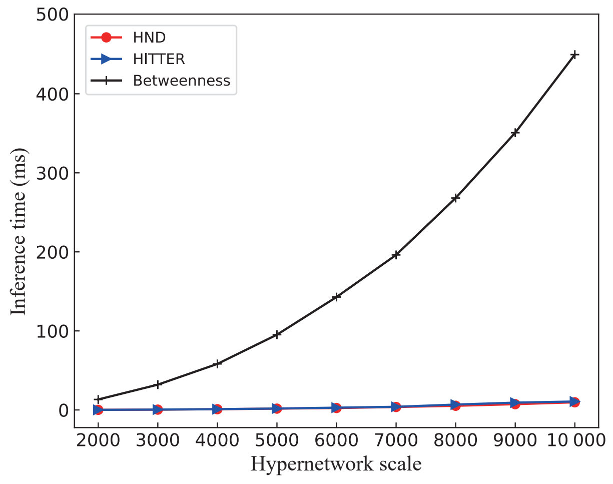


Fig. 5 Inference time under different hypernetwork scales.

Table 3 Efficiency comparison between HITTER and HND.

Method	Number of training iterations	Training time (s)
HITTER	84 766 (±17 470)	32 505.51 (±2808.82)
HND	101 (±45)	11 235.07 (±4866.57)

structure information in the node embeddings. Then, we train the model based on a large number of synthetic hypernetworks with the supervision of exact betweenness value in order to achieve real-world hypernetwork dismantling. Extensive experiments conducted on five real-world hypernetworks demonstrate that our proposed HND outperforms the baselines in terms of effectiveness and time efficiency. In our future work, we will consider the situation of noisy hyperedges in hypernetworks and design robust leaning-based hypernetwork dismantling methods accordingly.

Acknowledgment

This work was supported by the Anhui Province University Collaborative Innovation Project (No. GXXT-2022-091), the National Natural Science Foundation of China (No. 62006003), the Natural Science Foundation of Anhui Province (No. 2208085QF197), the Key Project of Nature Science Research for Universities of Anhui Province of China (Nos. 2022AH040019 and 2022AH05008637), the Hefei Key Common Technology Project (No. GJ2022GX15). The author is grateful to the Institute of Data Intelligence and Social Computing at Anhui University for funding this research.

References

[1] L. Qi, Y. Liu, Y. Zhang, X. Zhang, M. Bilal, and H. Song, Privacy-aware point-of-interest category recommendation in internet of things, *IEEE Internet of Things Journal*, vol. 9, no. 21, pp. 21398–21408, 2022.

[2] Y. Liu, D. Li, S. Wan, F. Wang, W. Dou, X. Xu, S. Li, R. Ma, and L. Qi, A long short-term memory-based model for greenhouse climate prediction, *International Journal of Intelligent Systems*, vol. 37, no. 1, pp. 135–151, 2022.

[3] Q. Wang, C. Zhu, Y. Zhang, H. Zhong, J. Zhong, V. S. Sheng, Short text topic learning using heterogeneous information network, *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 5, pp. 5269–5281, 2023.

[4] A. Braunstein, L. Dall'Asta, G. Semerjian, and L. Zdeborová, Network dismantling, *Proceedings of the National Academy of Sciences*, vol. 113, no. 44, pp. 12368–12373, 2016.

[5] M. Doostmohammadian, H. R. Rabiee, and U. A. Khan, Centrality-based epidemic control in complex social networks, *Social Network Analysis and Mining*, vol. 10, no. 1, pp. 32:1–32:11, 2020.

[6] D. Kempe, J. Kleinberg, and E. Tardos, Maximizing the spread of influence through a social network, in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC, USA, 2003, pp. 137–146.

[7] X.-L. Ren, N. Gleinig, D. Helbing, and N. Antulov-Fantulin, Generalized network dismantling, *Proceedings of the National Academy of Sciences*, vol. 116, no. 14, pp. 6554–6559, 2019.

[8] C. Fan, L. Zeng, Y. Sun, and Y.-Y. Liu, Finding key players in complex networks through deep reinforcement learning, *Nature Machine Intelligence*, vol. 2, pp. 317–324, 2020.

[9] F. Battiston, G. Cencetti, I. Iacopini, V. Latora, M. Lucas, A. Patania, J.-G. Young, and G. Petri, Networks beyond pairwise interactions: Structure and dynamics, *Physics Reports*, vol. 874, pp. 1–92, 2020.

[10] C. Fan, L. Zeng, Y. Ding, M. Chen, Y. Sun, and Z. Liu, Learning to identify high betweenness centrality nodes from scratch: A novel graph neural network approach, in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, New York, NY, USA, 2019, pp. 559–568.

[11] S. K. Maurya, X. Liu, and T. Murata, Graph neural networks for fast node ranking approximation, *ACM Transactions on Knowledge Discovery from Data*, vol. 15, no. 5, pp. 1556–4681, 2021.

[12] F. Morone and H. A. Makse, Influence maximization in complex networks through optimal percolation, *Nature*, vol. 524, pp. 65–68, 2015.

[13] L. Zdeborova, P. Zhang, and H.-J. Zhou, Fast and simple decycling and dismantling of networks, *Scientific Reports*, vol. 6, no. 1, p. 37954, 2016.

[14] S. Mugisha and H.-J. Zhou, Identifying optimal targets of network attack by belief propagation, *Physical Review E*, vol. 94, no. 1, p. 012305, 2016.

[15] M. Grassia, M. De Domenico, and G. Mangioni, Machine learning dismantling and early-warning signals of disintegration in complex systems, *Nature Communications*, vol. 12, no. 1, p. 5190, 2021.

[16] D. Yan, W. Xie, Y. Zhang, Q. He, and Y. Yang, Hypernetwork dismantling via deep reinforcement learning, *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 5, pp. 3302–3315, 2022.

[17] T. N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, in *Proc. 5th International Conference on Learning Representations*, Toulon, France, 2017, pp. 1–14.

[18] Y. Liu, H. Wu, K. Rezaee, M. R. Rezaee, O. I. Khalaf, A. A. Khan, D. Ramesh, and L. Qi, Interaction-enhanced and time-aware graph convolutional network for successive point-of-interest recommendation in traveling enterprises, *IEEE Transactions on Industrial Informatics*, vol. 19, no. 1, pp. 635–643, 2023.

[19] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, Graph attention networks, in *Proc. 6th International Conference on Learning Representations*, Vancouver, Canada, pp. 1–12, 2018.

[20] W. Hamilton, Z. Ying, and J. Leskovec, Inductive representation learning on large graphs, in *Proc. Advances in Neural Information Processing Systems*, Long Beach, CA, USA, 2017, pp. 1024–1034.

[21] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, Neural graph collaborative filtering, in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, Paris, France, 2019, pp. 165–174.

[22] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, Lightgcn: Simplifying and powering graph convolution network for recommendation, in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, Virtual Event, 2020, pp. 639–648.

[23] T. Chen and R. C.-W. Wong, Handling information loss of graph neural networks for session-based recommendation, in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Virtual Event, 2020, pp. 1172–1180.

[24] W. Chen, Y. Gu, Z. Ren, X. He, H. Xie, T. Guo, D. Yin, and Y. Zhang, Semi-supervised user profiling with heterogeneous graph attention networks, in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, Macao, China, 2019, pp. 2116–2122.

[25] D. Wang, P. Wang, K. Liu, Y. Zhou, C. E. Hughes, and Y. Fu, Reinforced imitative graph representation learning for mobile user profiling: An adversarial training perspective, in *Proceedings of the AAAI Conference on Artificial Intelligence*, Virtual Event, 2021, pp. 4410–4417.

[26] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, Hypergraph neural networks, in *Proceedings of the AAAI Conference on Artificial Intelligence*, Honolulu, HI, USA, 2019, pp. 3558–3565.

[27] N. Yadati, M. Nimishakavi, P. Yadav, V. Nitin, A. Louis, and P. Talukdar, HyperGCN: A new method for training graph convolutional networks on hypergraphs, in *Proc. Advances in Neural Information Processing Systems*, Vancouver, Canada, 2019, pp. 1511–1522.

[28] J. Huang and J. Yang, Unignn: A unified framework for graph and hypergraph neural networks, in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, Virtual Event, 2021, pp. 2563–2569.

[29] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, How powerful are graph neural networks? in *Proc. 7th International Conference on Learning Representations*, New Orleans, LA, USA, 2019, pp. 1−17.

[30] A. Bretto, *Hypergraph Theory*: *An Introduction*. Switzerland: Springer International Publishing, 2013.

[31] C. Berge, *Hypergraphs*: *Combinatorics of Finite Sets*. North Holland, Holland: Elsevier, 1989.

[32] M. T. Do, S.-E. Yoon, B. Hooi, and K. Shin, Structural patterns and generative models of real-world hypergraphs, in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Virtual Event, 2020, pp. 176−186.

[33] Y. Kook, J. Ko, and K. Shin, Evolution of real-world hypergraphs: Patterns and models without oracles, in *Proc. IEEE International Conference on Data Mining*, Sorrento, Italy, 2020, pp. 272–281.

[34] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, BPR: Bayesian personalized ranking from implicit feedback, in *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, Montreal, Canada, 2009, pp. 452–461.

[35] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, in *Proc. 3rd International Conference on Learning Representations*, San Diego, CA, USA, 2015, pp. 1−15.

[36] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-J. P. Hsu, and K. Wang, An overview of microsoft academic service (MAS) and applications, in *Proceedings of the 24th International Conference on World Wide Web*, Florence, Italy, 2015, pp. 243–246.

[37] A. R. Benson, R. Abebe, M. T. Schaub, A. Jadbabaie, and J. Kleinberg, Simplicial closure and higher-order link prediction, *Proceedings of the National Academy of Sciences*, vol. 115, no. 48, pp. E11221–E11230, 2018.

[38] A. Antelmi, G. Cordasco, C. Spagnuolo, and P. Szufel, Social influence maximization in hypergraphs, *Entropy*, vol. 23, no. 7, p. 796, 2021.

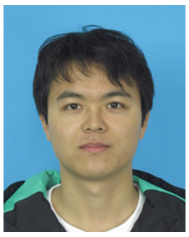
[39] C. M. Schneider, A. A. Moreira, J. S. Andrade, S. Havlin, and H. J. Herrmann, Mitigation of malicious attacks on networks, *Proceedings of the National Academy of Sciences*, vol. 108, no. 10, pp. 3838–3841, 2011.

[40] A.-L. Barabási and R. Albert, Emergence of scaling in random networks, *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

Yaguang Guo is the deputy director and associate researcher at E-government Research Institute, at School of Management, Hefei University of Technology, China. He received the BEng degree from Anhui University, China in 2000, and the MEng degree from Hefei University of Technology, China in 2005. He is currently a PhD candidate at School of Management, Hefei University of Technology, China. His current research interests include data governance, administrative approval reform, and Internet plus government services.

Wenxin Xie received the BEng degree from Liaoning Petrochemical University, China in 2018. He is currently a master student in computer science and technology at Anhui University, China. His main research interest is graph combinatorial optimization.

Qingren Wang is an associate professor at Global Cognition and International Communication Laboratory, Anhui University, China. He received the PhD degree from Hefei University of Technology, China. His current research interests include natural language processing, service computing, crowdsourcing, and data mining.


Dengcheng Yan received the BEng and PhD degrees from University of Science and Technology of China, China in 2011 and 2017, respectively. From 2017 to 2018, he was a core big data engineer at Research Institute of Big Data, iFlytek Co. Ltd., Hefei, China. He is currently a lecturer at Anhui University, China. His research interests include software engineering, recommendation systems, and complex networks.

Yiwen Zhang is a professor at School of Computer Science and Technology, Anhui University, China. He received the PhD degree from Hefei University of Technology, Hefei, China. His current research interests include personalized recommendation, edge computing, and data mining.